

Texture Regimes for Entropy-Based Multiscale Image Analysis

Sylvain Boltz^{1,2}, Frank Nielsen¹, and Stefano Soatto¹

¹ Laboratoire d'Informatique, École Polytechnique, 91128 Palaiseau Cedex, France
`{boltz,nielsen}@lix.polytechnique.fr`

² UCLA Vision Lab, University of California, Los Angeles; Los Angeles – CA, 90095
`soatto@ucla.edu`

Abstract. We present an approach to multiscale image analysis. It hinges on an operative definition of texture that involves a “small region”, where some (unknown) statistic is aggregated, and a “large region” within which it is stationary. At each point, multiple small and large regions co-exist at multiple scales, as image structures are pooled by the scaling and quantization process to form “textures” and then transitions between textures define again “structures.” We present a technique to learn and agglomerate sparse bases at multiple scales. To do so efficiently, we propose an analysis of cluster statistics after a clustering step is performed, and a new clustering method with linear-time performance. In both cases, we can infer all the “small” and “large” regions at multiple scale in one shot.

1 Introduction

Textures represent an important component of image analysis, which in turn is useful to perform visual decision tasks – such as detection, localization, recognition and categorization – efficiently by minimizing decision-time complexity [12]. The goal of image analysis¹ is to compute *statistics* (deterministic functions of the data) that are at the same time insensitive to nuisance factors (*e.g.*, view-point, illumination, occlusions, quantization) and useful to the task (*i.e.*, in the context of visual decision tasks, discriminative). Such statistics are often called *features*, or *structures*. Such structures have to satisfy a number of properties to be useful, such as *structural stability*, *commutativity*, and *proper sampling* [12]. The *dual* of such structures, in a sense made precise by Theorem 5 of [12], are textures, or more precisely *stochastic textures*, defined by spatial stationarity of *some* (a-priori unknown) statistics. *Regular textures*, on the other hand, are defined by cyclo-stationarity (or stationarity with respect to a discrete group) of *some* (a-priori unknown) structure.

¹ Image analysis refers to the process of “breaking down the image into pieces,” which is *prima facie* un-necessary and even detrimental for data storage and transmission tasks ([11], p. 88), but instead plays a critical role in visual decision tasks [12].

Whether a region of an image is classified as texture or structure *depends on scale*.² A region can be a structure at some scale, a texture at a coarser scale, then again a structure at yet a coarser scale and so on (Fig. 1), reflecting the lack of *causality* in two-dimensional scale-space [9]. Therefore, we do not seek for a *single* transition from structure to texture [17,18], but instead seek to represent the entire phase-space of transitions at each location in an image.

To address these issues, in Sect. 2 we introduce a definition of texture that guides the development of efficient schemes for multiscale coding in Sect. 3, where we introduce an algorithm to compute statistics based on three fast clustering algorithms reviewed in Sect. 4.1 and 4.3, and a new variant introduced in Sect. 4.2. These clustering algorithms allow us to perform multiscale analysis in one shot (Sect. 5). The analysis can also be done directly in the clustering process with linear complexity (Sect. 5.2).

Our characterization of textures uses three ingredients: A statistic, ψ , the minimal domain where such a statistic is pooled, ω , and the maximal domain (in the sense of inclusion) where it is stationary, Ω . Therefore, we focus on defining suitable classes of statistics, and on designing efficient algorithms to partition the image into multiple regions. This is done through efficient techniques to create sparse bases (dictionaries), using clustering and dimensionality reduction in high-dimensional non-Euclidean spaces. In particular, we introduce “kNN-Quick Shift,” a generalization of [16] modified to handle data distributed in high-dimensional spaces, and to allow different variables, such as scale, as “gap” measures. This enables simultaneous estimation of “small” ω and “big” Ω regions can be performed by alternating Min-Max entropy segmentation in linear time. Our entropy measure exhibits a “staircase-like” behaviour, with each step determining the small regions ω at the lower edge, and the big regions at the upper edge. Note that we achieve this in one-shot, for all scales, without having to match patches or searching for periodic patterns [6].

2 Texture/Structure Multiple Transitions

Image structures are regions of the image that are *salient* (*i.e.*, critical points of some functional, Def. 4 of [12]), *repeatable* (*i.e.*, the functional is structurally stable, Def. 9 of [12]), and *insensitive* to nuisance factors (*i.e.*, the functional is invariant to group nuisances and commutes with respect to non-invertible ones, Def. 6 of [12]). For zero-dimensional structures (attributed points, or *frames*), it has been shown that the attributed Reeb tree (ART) is a complete invariant with respect to viewpoint and contrast away from occlusions [13]. However, *occlusions* of viewpoint and illumination (cast shadows) yield *one-dimensional structures*, such as edges and ridges. The main technical and conceptual problem in encoding

² Note that scaling alone is not what is critical here. Scaling is a group, so one can always represent any orbit with a single canonical element. What is critical is the composition of scaling with *quantization*, which makes for a *semi-group*. Without quantization we would not need a notion of stochastic texture, since any region would reveal some structure at a sufficiently small scale.

them is that such *critical structures* (extrema and discontinuities) *are not defined in a digital image*. For this reason, we must first *define* a notion of “discrete continuity,” lest every pixel boundary is a structure. This is achieved by designing a *detector*, usually an operator defined on a scalable domain, and searching for its extrema at each location in the image, at each scale. In principle one would have to store the response of such detectors at all possible scales. In practice, owing to the statistics of natural images, we can expect the detector functional to have isolated critical loci that can be stored in lieu of the entire scale-space. In between critical scales, structures become part of aggregate statistics that we call *textures*.

To make this more precise, we define a texture as a region $\Omega \subset D$ within which *some* image statistic ψ , aggregated on a subset $\omega \subset \Omega$, is spatially stationary.³ Thus a texture is defined by two (unknown) regions, *small* ω and *big* Ω , an (unknown) statistic $\psi_\omega(I) \doteq \psi(\{I(y), y \in \omega\})$, under the following conditions of *stationarity* and *non-triviality*:

$$\psi_\omega(I(x+v)) = \psi_\omega(I(x)), \quad \forall v \mid x \in \omega \Rightarrow x+v \in \Omega \quad (1)$$

$$\Omega' \setminus \omega \neq \emptyset \Rightarrow \psi_{\Omega'}(I) \neq \psi_\Omega(I). \quad (2)$$

The small region ω , that defines the intrinsic scale $s = |\omega|$ (the area of ω), is minimal in the sense of inclusion⁴. Note that, by definition, $\psi_\omega(I) = \psi_\Omega(I)$. A texture segmentation is thus defined, for every quantization scale s , as the solution of the following optimization with respect to the unknowns $\{\Omega_i\}_{i=1}^N, \{\omega_i\}_{i=1}^N, \{\psi_i\}_{i=1}^N$

$$\min \sum_{i=1}^{N(s)} \int_{\Omega_i} d(\psi_{\omega_i}(I(x)), \psi_i) dx + \Gamma(\Omega_i, \omega_i) \quad (3)$$

where Γ denotes a regularization functional and d denotes a distance, for instance ℓ^2 , or a nonparametric divergence functional [2,8].

In Sect. 3 we discuss the role of the statistics ψ . In Sect. 4, we discuss about some clustering algorithms, introducing along the way a novel extension of a clustering algorithm that is suited for high-dimensional spaces. Finally we build on these clustering algorithms to derive, in Sect. 5, two methods to automatically compute the set of all regions $\{\omega_i\}$ and $\{\Omega_i\}$.

3 Multiscale Feature Selection and Dictionary Agglomeration

The difficulty in instantiating the definition of texture into an algorithm for image analysis is that neither the regions ω_i, Ω_i , nor the statistics ψ_ω are known

³ Constant-color regions are a particular (trivial) case of texture, where the statistic $\psi(I) = I$ is pooled on the pixel region $\omega = \{x\}$. It is an unfortunate semantic coincidence that such regions are sometimes colloquially referred to as “textureless.”

⁴ $\{I(x), x \in \omega\}$ is sometimes called a *texton* [7], or *texture generator*. This definition applies to both “periodic” or “stochastic” textures. Regions with homogeneous color or gray-level are a particular case whereby ω is a pixel, and do not need separate treatment.

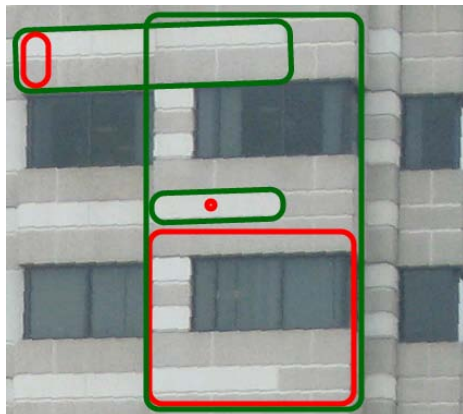


Fig. 1. Left: multiple Texture/Structure Transitions: The same point can be interpreted as either structure or texture depending on scale. Starting from a small red region ω , the green region Ω determines the domain where some statistic computed in ω is stationary (relative some group, which includes cyclo-stationarity when the group is discrete).

a-priori. It is therefore common to define ψ in terms of a class of functions such as a Gabor wavelets or other bases learned directly from the image under sparsity constraints [1]. One can even consider just samples of the image in a window of varying size around each pixel [15]. Whatever representation one chooses for a local neighborhood of the image at a given scale, the fact that all points have to be represented at all scales causes an explosion of complexity. This can be mitigated by clustering the dictionary elements into a codebook, with each dictionary element encoded with an index and representing a mode in the data distribution. Each image region is then then represented by an histogram of these indices. In principle, one could take these to be our statistics ψ , and represent *structures* as locations where the label histogram is surrounded by different ones, and *textures* as locations where the label histogram is surrounded by similar ones.

However, the dictionaries thus learned are usually very large and cumbersome to work with. Therefore, one can reduce the dimensionality of the representation by reducing the number of atoms in the dictionary. However, in order to achieve the *insensitivity* to nuisance factors described in the previous section, clustering cannot just be performed with respect to the standard ℓ_2 distance as the atoms may undergo deformations. Clustering with an histogram-based distance is also ill-advised as the distributions of different classes have significant overlap. Kullback-Leibler's divergence naturally adapts to the supports and the modes of the distributions, and is therefore a natural choice for divergence measure. The feature space is then chosen so as to discount nuisance variability.

We propose clustering in three different feature spaces to agglomerate patches modulo three different types of transformation (Fig. 2): First, we consider $\psi_\omega \doteq \{I(x, y), x, y\}_{(x, y) \in \omega}$ to get rid of small translations. In this feature space one

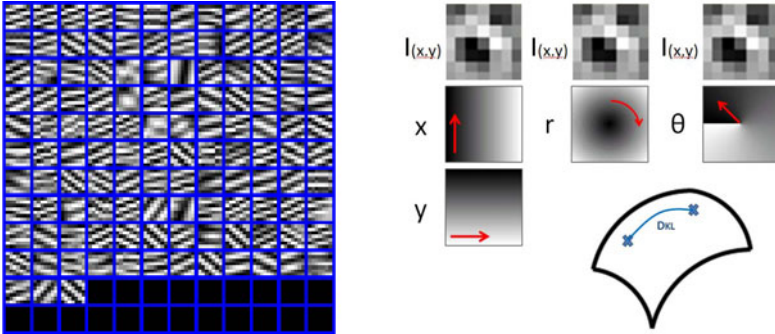


Fig. 2. (left) Clustering with a “bag-of-features” dictionary of 256 image patches. Patches identical modulo small feature / geometry deformations are now agglomerated to one exemplar texture patch. (right) Clustering on sparse representation of images with uncertainty on value and position: the three columns show the different feature spaces, and red arrow shows the direction along which neighbors are preferably searched for. The distance on these feature spaces is the symmetric Kullback-Leibler divergence.

pays a small price to align two similar $I(x, y)$ as long as their (x, y) distance is small. Similarly, polar coordinates $\{I(x, y), r, \epsilon, \theta\}$, with a small weight ϵ on the angle, are insensitive to small rotations and with a small weight on the radius $\{I(x, y), \epsilon, r, \theta\}$ they are insensitive to small scalings (Fig. 2).

Now, the last step is the most critical, for it involves clustering in high-dimensional and highly non-Euclidean spaces (Fig. 2). We will describe our approach in the next two sections.

4 Three Fast Clustering Algorithms

In this section we use two existing clustering algorithms, Quick Shift (QS) and Statistical Region Merging (SRM), and introduce a novel one, “kNN Quick Shift,” that adapts QS to high-dimensional data.⁵ The purpose is to show that the analysis that follows is not dependent on the particular algorithm to arrive at a clustering tree.

4.1 Mode Seeking with Quick Shift

Quick Shift [16] is a modification of Medoid-Shift that retains its benefit of operating on non-Euclidean spaces and still converges in one iteration.

$$y_i(1) = \arg \min_{j: P(j) > P(i)} D_{ij} \quad (4)$$

$$\tau(i) = D_{iy_i(1)} \quad (5)$$

⁵ Those two families of algorithms were also recently combined in [4].

Its main advantage is simplicity and speed. One clustering with Quick Shift gives a full segmentation tree as all nodes are connected to each other with a different strength τ called the *gap*. Thresholding this gap with different values thus yields different segmentations. The most common use of these clustering algorithms is with the feature space $\{I(x), x, y\}$, yielding compact regions of uniform luminance or color usually called *superpixels*. In practice, the full matrix D_{ij} does not need to be built, as the feature space often has a geometric component, so physical neighbors are also neighbors in feature space. This limits the search to a small local window. Even if no geometric prior is available, one can still use a window of a certain size h around each datum. When the data distribution is high-dimensional and sparsely distributed, a large h has to be selected, leading to oversmoothing the estimate of the probability density function (PDF), and to a computationally intensive search. In the next section we introduce a modification of this algorithm designed to mitigate this problem, similarly to what [5] has done for Mean Shift.

4.2 kNN Quick Shift

To extend QS to high-dimensional data spaces we replace the Parzen density estimator with a balloon estimator. The analysis of [14] shows that, although balloon estimators underperform Parzen in one dimension, they improve as the dimension of the space increases. We choose the neighborhood of possible connections to be the k -nearest neighbors $\mathcal{N}_k(i)$ of each point.

$$y_i(1) = \arg \min_{j: j \in \mathcal{N}_k(i) \& P_j > P_i} D_{ij} \quad (6)$$

The resulting kNN Quick Shift is made very fast by using approximate nearest neighbors with $(1 + \epsilon)$ tolerance. In practice, this works well if k is low, so we implemented a recursive kNN-Quick Shift algorithm: It first builds a tree connecting pixel values, then unconnected superpixels are connected until every node is linked.

When clustering pixels in an image, D_{ij} is simply the Euclidean distance between two pixel features. When clustering patches for agglomeration of bases modulo some deformations, D_{ij} is the symmetric Kullback-Leibler divergence computed on three different feature spaces (Sec. 3). The parameter k exerts direct control on the cluster size. It can therefore be used as a gap measure to perform a cut of the tree structure provided by QS. This is particularly relevant in the context of texture analysis, where we seek the smallest ω and largest Ω regions where certain statistics are stationary. However, it is inconsistent with a region based energy as defined in eq. (3).

4.3 A Fast Statistical Region Merging (SRM)

SRM is an efficient greedy algorithm [10] for region merging with theoretical guarantees. Every pair of adjacent pixels (both horizontally and vertically) is

assigned a strength value, for instance the absolute value of their intensity difference. The list is then sorted, and location labels retained. For each pair of pixels, a test called predicate is run to decide if the regions are to be merged. This runs in linear time as it only goes through all the pixel pairs once ($2N$ -complexity). The region merging structure is a union find data structure which allows finding pixel labels with complexity $\mathcal{O}(1)$. In order to build a segmentation tree with this algorithm, the predicate is made to depend on a scalar parameter, and the same algorithm is run repeatedly for increasing values of the parameter and with pairs of adjacent regions instead of pairs of pixels.

5 Recursive Max-Min Entropy for Texture Analysis

From the operational definition of texture (2), we seek to efficiently compute a multiscale representation to simultaneously detect the small ω and large Ω . The basic intuition comes from the observation that aggregating adjacent superpixels yields an increase in the entropy of the aggregate descriptor, up to the point where a minimum stationary region is reached, ω . At that point, aggregating further regions will not change the entropy, because of the stationarity assumption (of course, the complexity of the encoding will decrease, as more and more superpixels are lumped into the same region), up to the point where the boundary of the large region Ω is reached. Aggregating superpixels outside this region will cause the entropy to resume its climb.

The recursive functional reads, initializing $\omega_i^{(0)}$ as N different regions of 1 pixel size, where N is the number of pixels in the image,

$$\begin{cases} \Omega_i^{(k)} = \arg \min_{\Omega_i} \sum_{i=1}^{N(s)} \int_{\Omega_i} \mathcal{H}(\psi_{\omega_i^{(k)}}(I(x))) dx + \Gamma(\Omega_i) \\ \omega_i^{(k)} = \arg \max_{\omega_i} \sum_{i=1}^{N(s)} \int_{\omega_i} \mathcal{H}(\psi_{\Omega_i^{(k-1)}}(I(x))) dx - \Gamma(\omega_i) \end{cases} \quad (7)$$

where $\mathcal{H}$ is the Shannon entropy.

We propose two methods to perform this optimization. Building from a segmentation trees e.g. Sec. 4.1, 4.2, or 4.3 Method 1 performs a constrained optimization as a line search in the tree of superpixels. While Method 2 is a free-form optimization in the image domain.

5.1 Method 1: Constrained Solution from a Pre-processed Segmentation Tree

To instantiate this, we use the entropy-based saliency function introduced in [8], followed by entropy-based segmentation, as customary [2].

Using the dictionary features defined in Sect. 3, and the superpixel segmentation map at a given scale, one can compute an entropy $\mathcal{H}$ of features inside the superpixel containing i :

$$\mathcal{H}_i(s) = - \sum_{x=1}^D P_i(x, s) \log P_i(x, s) \quad (8)$$

where P_i is the distribution of the reduced dictionary features built on the region defined by i , *i.e.*, the superpixel $\mathcal{S}(i, s)$, D is the size of the dictionary,

$$P_i(x) = \frac{1}{|\mathcal{S}(i, s)|} \sum_{p \in \mathcal{S}(i, s)} \delta(x - d(p)) \quad (9)$$

and δ is Dirac's delta, $d(p)$ is the index of the dictionary at point p . The small scale of a texture is defined as the largest scale at which entropy stops increasing:

$$\begin{cases} \omega(i, s) = \mathcal{S}(i, s') \\ s' = \arg \max_{t > s} \{t \mid \forall v \ s \leq v \leq t, \frac{d\mathcal{H}_i(v)}{ds} > 0\}. \end{cases} \quad (10)$$

The stationary domain of the texture Ω is simply defined as the boundary of the region past which entropy resumes increasing,

$$\begin{cases} \Omega(i, s) = \mathcal{S}(i, s'') \\ s'' = \arg \max_{t > s} \{t \mid \forall v \ s \leq v \leq t, \frac{d\mathcal{H}_i(v)}{ds} \leq 0\} \end{cases} \quad (11)$$

Therefore, we perform the final segmentation at the maximum region that preserves stationarity.

While this method can be used with any segmentation tree, the solution will be constrained as unions of preprocessed segmentations. An unconstrained free-form solution can be found by building a segmentation tree that optimizes directly a Min-Max entropy in linear time using the properties of SRM.

5.2 Method 2: SRM with Alternate Min-Max Entropies

We start from an initial segmentation ω_i , *e.g.*, from SRM, to initialize the statistics, then perform SRM again with the ordering of neighboring segments given by sorting the strength between segments $\Gamma = \int_{\partial\Omega_i} \|\nabla I(x)\|^2 dx$ in increasing order. Now the predicate changes to an entropy-increasing or -decreasing test, and regions are merged only if entropy keeps decreasing or is constant. Once a region Ω_i is found, we turn to maximizing the entropy to find the region ω_i . The ordering of Γ neighboring segment is now sorted by decreasing order, and the regions are merged if entropy increases. In this way, one can define an alternating Min-Max entropy exploration of the image with the same complexity, since the neighboring graphs are only processed once. The number of merging tests is again linear in the number of pixels in the image.

5.3 Features Persistence and Stability

If the hypothesis underlying our definition of texture is correct, entropy will have a staircase-like behavior, with flat plateaus bounded below (in the sense

of inclusion) by the small region ω , and above by Ω . As the same region can switch back-and-forth from texture to structure, we expect several such plateaus as the scale of inclusion changes. In the next section, we verify this hypothesis empirically on different superpixels from natural images. While this behavior is given *a priori* in the Min-Max entropy clustering, it is not obvious that it will be manifest when using any segmentation tree.

Those entropy profiles at each pixel are now agglomerated into one global histogram of entropies using a voting approach. As a staircase value of entropy appears at one pixel, it sums as a weighted contribution in the histogram of entropies, the weight being simply the length of the step. This histogram thus shows the different stable regimes of entropies appearing in the image. Knowing this histogram, one can deduce simply the local scale at a pixel position by doing mode seeking on this histogram (smoothed as a PDF). The definition of a local scale at one pixel position is the smallest scale where a mode value of the global histogram entropy appears. This allow us to perform stable segmentation and description of the natural scale of the image at a pixel, according to a structural stability criterion where the length of each step measures the *structural stability margin* [12].

6 Experimental Results

6.1 Computational Speed

In this section we explore the complexity and performance of the one dictionary learning and four clustering algorithms discussed. Computational speed is shown in Table 1, measured in seconds on a matlab/C implementation. Three different methods are used to build the dictionaries : (1) Color dictionary using k-means (2) Texture dictionary using k-means (3) Texture dictionary using k-means and agglomerated using a QS with Kullback-Leibler divergence. Based on those three different features, four different segmentation trees have been built: (a) Classical QS as described in Sect. 4.1 (b) kNN QS , with a scale parameter, designed for high dimensional spaces in Sect. 4.2 (c) Classical SRM as described in Sect. 4.3 (d) SRM with alternated min max entropies as described in Sect. 5.2

Table 1. Running times in seconds, (1,2,3) dictionary learning methods (a,b,c,d) segmentation tree methods

Method	(1)	(2)	(3)	(a)	(b)	(c)	(d)
Speed	0.2	4.1	8.1	0.4	0.1	1.5	3.8

6.2 Dictionary Agglomeration

We illustrate agglomeration by clustering a dictionary built on the “Barbara” image to eliminate nuisance variations such as small rotations, translations, and contrast changes. QS does not require a smooth embedding, so it can be used with

a non-Euclidean metric, for instance one defined on the quotient space under the nuisance group. We use the symmetrized Kullback-Leibler divergence estimated with an efficient kNN-based estimator [3]. For every atom we build the pairwise distance matrix D_{ij} in (6). The first stage, with feature space $\{I(x, y), x, y\}$, forms a big cluster containing most of the texture elements “stripes” (Fig. 2). This dictionary now contains only one atom representing this texture cluster, or “exemplar.”

In order to evaluate the efficiency of this agglomeration we take 32 random images from the Berkeley segmentation dataset. For each one we compare four ways of building a 128-atom dictionary: (i) direct k-means on the patches, (ii) first learning 256 clusters, then reducing QS using either ℓ_2 , or (iii) KL clustering on $\{I(x, y)\}$, and finally (iv) KL clustering on $\{I(x, y), x, y\}$. One measure of efficiency of these dictionaries is the spatial coherence of the index of the atoms used. To measure it, one can compute first a color segmentation (in order to be independent of the texture measures) on each image and sum the entropies of each segment. The lower this entropy $\mathcal{H}_{\text{average}}$, the more coherent the index of the atoms.

$$\mathcal{H}_{\text{average}}(\text{feature}) = \frac{1}{32} \sum_{i=1}^{32} \frac{1}{N_S(i)} \sum_{s=1}^{N_S(i)} \mathcal{H}_{(\text{feature})}(i, s) \quad (12)$$

where $N_S(i)$ is the number of superpixels in image i across all scales, $\mathcal{H}(\text{feature})(i, s)$ is the entropy of a given feature, in image i , inside superpixel s . In this section the feature used is the index of the dictionary. Average entropies shown in Table reveal that one can obtain coherent sparse decompositions in natural images and thus efficient dimensionality reduction.

Table 2. Agglomeration of dictionaries for efficient sparse representation. Sum of entropies over all the superpixels. (a) k-means on 128 elements, (b) (c) (d) k-means with an initial size of 256 reduced to 128 with, (b) ℓ_2 clustering, (c) KL clustering on $\{I(x, y)\}$ (d) KL clustering on $\{I(x, y), x, y\}$.

	(a)	(b)	(c)	(d)
$\mathcal{H}_{\text{average}}(\text{index})$	3.25	3.10	3.17	2.21

This method is also computationally tractable as it runs on the space of bases, rather than the space of all image patches as in [2]. However, as the scale of the texture is unknown, it can contain many dictionary elements. A solution is to look for the dictionary dimension that gives uniform regions in the space of coefficients. Another solution is to try to find the natural scale of the textures using region growing algorithms, in our case superpixels aggregating across the tree of possible segmentations.

6.3 Multiscale Region Analysis

An illustration of the multiscale region analysis is shown in Figure 3. From one image, a superpixel map is computed at different scales. Then the dictionary features are computed and agglomerated. Finally, for six randomly select points inside each superpixel, we plot the variations of entropy. It is evident that, as scale increase, entropy increases in steps. The stationary regime corresponds with superpixels merging with others of similar distribution. Structural transitions occur when superpixels merge that have different distributions. The process ends when no new regions are discovered. These phase changes serve to detect ω_i and Ω_i , which are displayed for different superpixels at key scales when the entropy regime changes. In Figure 4, critical scales at successive levels are shown. First on a synthetic image, starting from one pixel, critical scales are: the pixel itself, the dark brown region minimizing entropy, then maximizing entropy, the light brown region to form an "L", then minimizing entropy again, until the entire image is segmented. The same process is shown for a natural image; a light brick is first selected, then agglomerated with a darker one, then with a window, then with the whole building since all the statistics that describe the building are captured.



Fig. 3. It shows the entropy regime of 6 randomly selected points in the first image, by going through the different scales of the segmentation tree. Staircase is visible and shows the successive entropy regime of superpixels from successive ω regions to Ω regions. The regions are computed with QS trees. Detection of critical scales of textures on two different images. The segmentation scheme is now SRM Min-Max.

6.4 Scale Segmentation

The natural scale of a pixel is then extracted as described in Sect. 5.3. We use the segmentation tree SRM Min-Max with the features based on agglomerated dictionaries. Once the critical scales are computed, one builds a PDF of entropies over all superpixels. The scale at one pixel is defined as the smallest scale at which a mode appears. Those modes show stable regimes of entropies and can be used as a feature for scale segmentation. They also have the property of being accurate at boundaries, since the size of statistics is adaptive.

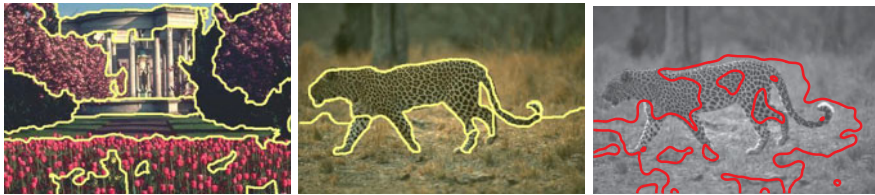


Fig. 4. Scale segmentation results on two images. By computing the statistics at the right scale, one can segment boundaries with pin-point precision, rather than suffering from “fat-boundary effects” common in texture segmentation. Last image shows a result with a standard texture segmentation algorithm [6] suffering from uniform scale selection and “fat-boundary effects”

6.5 Stability

To evaluate stability, again thirty-two images of the Berkeley segmentation dataset are again randomly selected. If the critical scales extracted are correct, there should be some coherence in all the regions extracted across the image (since our definition does not leverage on any matching, this condition is not forced by construction). The measure of stability is then the average entropy over the 32 images over all the superpixels as described in Sect. 6.2 and Equation (12). The features used here are size and color of the superpixels. Such features, if regions are stable and consistent, should have a low entropy across the image. That means that many regions should be similar in size and color in natural images.

Table 3. Stability of critical scales extracted using four different segmentation trees (a,b,c,d) based on three different features (1,2,3). The dictionaries, all of size 128, are: (1) Color (2) Texture (3) Agglomerated Texture using QS with Kullback-Leibler as described in Sect. 3. The segmentation trees are (a) QS described in Sect. 4.1 (b) kNN QS described in Sect. 4.2 (c) SRM described in Sect. 4.3 (d) SRM Min-Max entropy described in Sect. 5.2. The best result for each features and entropy (color or size of the critical scales) is shown in bold. The best overall result is obtained with SRM Min-Max with agglomerated texture dictionary features.

	(a-1)	(a-2)	(a-3)	(b-1)	(b-2)	(b-3)
$\mathcal{H}_{\text{average}}(\text{color})$	3.17	3.01	2.88	3.08	2.97	2.81
$\mathcal{H}_{\text{average}}(\text{size})$	4.87	4.52	4.18	4.15	4.12	4.05
	(c-1)	(c-2)	(c-3)	(d-1)	(d-2)	(d-3)
$\mathcal{H}_{\text{average}}(\text{color})$	3.11	3.01	2.95	2.22	2.14	2.07
$\mathcal{H}_{\text{average}}(\text{size})$	4.51	4.21	4.12	4.17	4.08	3.98

7 Discussion

We have presented an approach to multiscale texture analysis. The operative definition of texture we introduce guides the development of algorithms that efficiently enable the estimation of all the “small regions” (a.k.a. “texton regions”), the “big regions” (a.k.a. “texture segments”) and the statistics within. We have introduced a novel clustering algorithm adapted for high-dimensional spaces, and showed how an information-theoretic criterion can be used to define the “gaps” to simultaneously detect small and large regions.

Acknowledgment

Research supported by ONR N00014-08-1-0414 and ARO 56765-CI.

References

1. Aharon, M., Elad, M., Bruckstein, A.M.: The k-svd: An algorithm for designing of overcomplete dictionaries for sparse representation. *IEEE Transactions On Signal Processing* 54(11), 4311–4322 (2006)
2. Awate, S.P., Tasdizen, T., Whitaker, R.T.: Unsupervised texture segmentation with nonparametric neighborhood statistics. In: *European Conference on Computer Vision*, Graz, Austria, pp. 494–507 (2006)
3. Boltz, S., Debreuve, E., Barlaud, M.: High-dimensional statistical distance for region-of-interest tracking: Application to combining a soft geometric constraint with radiometry. In: *IEEE International Conference on Computer Vision and Pattern Recognition*, Minneapolis, USA (2007)
4. Chazal, F., Guibas, L.J., Oudot, S.Y., Skraba, P.: Persistence-based clustering in Riemannian manifolds. *Research Report 6968*, INRIA (June 2009)
5. Georgescu, B., Shimshoni, I., Meer, P.: Mean shift based clustering in high dimensions: A texture classification example. In: *IEEE International Conference on Computer Vision*, p. 456 (2003)
6. Hong, B.W., Soatto, S., Ni, K., Chan, T.F.: The scale of a texture and its application to segmentation. In: *IEEE International Conference on Computer Vision and Pattern Recognition* (2008)
7. Julesz, B.: Textons, the elements of texture perception and their interactions. *Nature* (1981)
8. Kadir, T., Zisserman, A., Brady, M.: An affine invariant salient region detector. In: *European Conference on Computer Vision* (2004)
9. Lindeberg, T.: *Scale-space theory in computer vision*. Kluwer Academic, Dordrecht (1994)
10. Nock, R., Nielsen, F.: Statistical region merging. *IEEE Transactions Pattern Analysis Machine Intelligence* 26(11), 1452–1458 (2004)
11. Robert, C.P.: *The Bayesian Choice*. Springer, New York (2001)
12. Soatto, S.: Towards a mathematical theory of visual information (2010) (preprint)
13. Sundaramoorthi, G., Petersen, P., Varadarajan, V.S., Soatto, S.: On the set of images modulo viewpoint and contrast changes. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (June 2009)

14. Terrell, G.R., Scott, D.W.: Variable kernel density estimation. *The Annals of Statistics* 20, 1236–1265 (1992)
15. Varma, M., Zisserman, A.: A statistical approach to material classification using image patch exemplars. In: *IEEE Transactions Pattern Analysis Machine Intelligence* (to appear)
16. Vedaldi, A., Soatto, S.: Quick shift and kernel methods for mode seeking. In: *European Conference on Computer Vision*, vol. IV, pp. 705–718 (2008)
17. Wu, Y.N., Guo, C., Zhu, S.C.: Perceptual scaling. *Applied Bayesian Modeling and Causal Inference from an Incomplete Data Perspective* (2004)
18. Zhu, S.C., Wu, Y.N., Mumford, D.: Minimax entropy principle and its application to texture modeling. *Neural Computation* 9, 1627–1660 (1997)